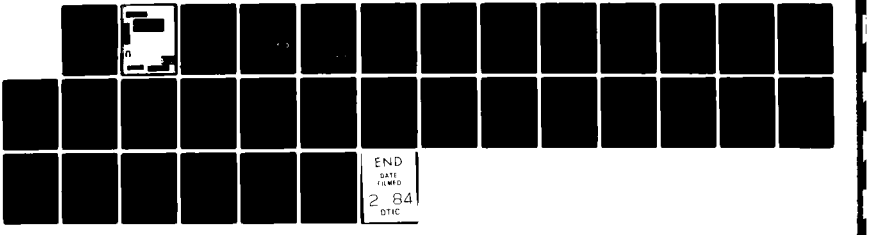
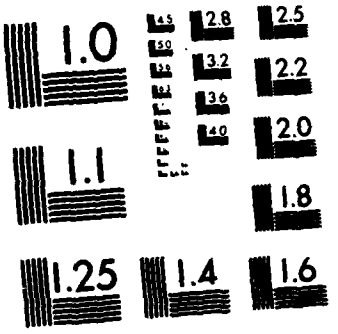
AD-A137 355 AUTOMATIC PLANNING OF FINE MOTIONS: CORRECTNESS AND COMPLETENESS(U) CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST M T MASON 14 DEC 83 CMU-RI-TR-83-18 1/1

UNCLASSIFIED F/G 9/2 NL



END
DATE FILMED
2 84
DTIC



Carnegie-Mellon University

The Robotics Institute

Technical Report

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER C MU-RI-TR-83-18	2. GOVT ACCESSION NO. AD-A137 355	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AUTOMATIC PLANNING OF FINE MOTIONS: CORRECTNESS AND COMPLETENESS		5. TYPE OF REPORT & PERIOD COVERED Interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Matthew T. Mason		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University The Robotics Institute Pittsburgh, PA. 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December 1983 13. NUMBER OF PAGES 26
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this paper we explore a method for automatic planning of robot fine-motion programs, first described in [Lozano-Perez, Mason, and Taylor 1983]. The primary result is a variation that is shown to be "bounded-complete"--the method obtains a solution whenever a solution consisting of a bounded number of motions exists.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Automatic Planning of Fine Motions: Correctness and Completeness

Matthew T. Mason

CMU-RI-TR-83-18

Computer Science Department
and Robotics Institute

Carnegie-Mellon University

Pittsburgh, Pennsylvania 15213

14 December 1983

To be presented at the IEEE Computer Society International Conference on Robotics, March 13-15, 1984, Atlanta, Georgia. This research was supported by the Robotics Institute and the Computer Science Department, Carnegie-Mellon University.

Table of Contents

List of Figures

ABSTRACT

In this paper we explores a method for automatic planning of robot fine-motion programs, first described in [Lozano-Pérez, Mason, and Taylor 1983]. The primary result is a variation that is shown to be "bounded-complete" — the method obtains a solution whenever a solution consisting of a bounded number of motions exists.

I. Introduction.

By the clever use of task geometry and sensory information, a manipulator can often perform tasks exceeding the nominal accuracy of the manipulator. Unfortunately, such strategies can be difficult to discover and must be constructed anew for each task. In this paper we explore an approach to automatic planning of such strategies, first described in [Lozano-Pérez, Mason, and Taylor 1983].

Figure 1 shows some simple fine-motion strategies used to insert a two-dimensional peg in a hole. The motions shown in these examples do not rely on the positioning capability of the controller, that is, they do not try to attain a particular location based on the position encoders in the arm. Rather, they try to move in a general direction, using contact with objects to guide the motion to specific goals. An important dividend of this approach is that it is less vulnerable to control error and to variations in the task environment.

Automatic planning of fine-motions will require some ability to reason about contact situations, and to reason about situations involving uncertainty and control error. In section I.A we will briefly describe techniques used for modeling and reasoning about control, motion, contact, sensory error, and control error. In section II we provide a detailed description of the procedure FMP, as well as some variations. In section III we present a proof that FMP is correct and bounded-complete—that is, FMP obtains a solution whenever a solution consisting of a bounded number of motions exists.

Example.

The planning process for moving a point into a hole in two dimensions is illustrated in figure 2. The task is to move the point to goal G at the bottom of the hole. We represent uncertainty in the initial position of the point by saying only that it must be somewhere in the set I. A good strategy must be successful no matter where in I the true position lies. A direct motion towards the bottom of the hole will not work if the point's initial position is near the left edge or the right edge of I. A better strategy is to use two motions: the first motion deliberately strikes the surface to one side of the hole; and the second motion slides back towards the hole, allowing the point to fall in when it crosses the edge of the hole. This is an instance of the biased search strategy illustrated in figure 1.

It is difficult to reason about a sequence of motions, so we will use an approach which allows us to reason about a single motion at a time. The basic idea will be familiar to those who are acquainted with dynamic programming, or with traditional artificial intelligence planning techniques, where it is known as backward-chaining, without back-tracking. If a sequence of N motion commands $\{s^i\}$ reliably moves the robot from I to G, then there is a set R such that the first $N-1$ motions will reliably move the robot from I to R, and the last motion will reliably move the robot from R to G. Using this observation, a sequence-planning problem can be reduced to N single-motion-planning problems.

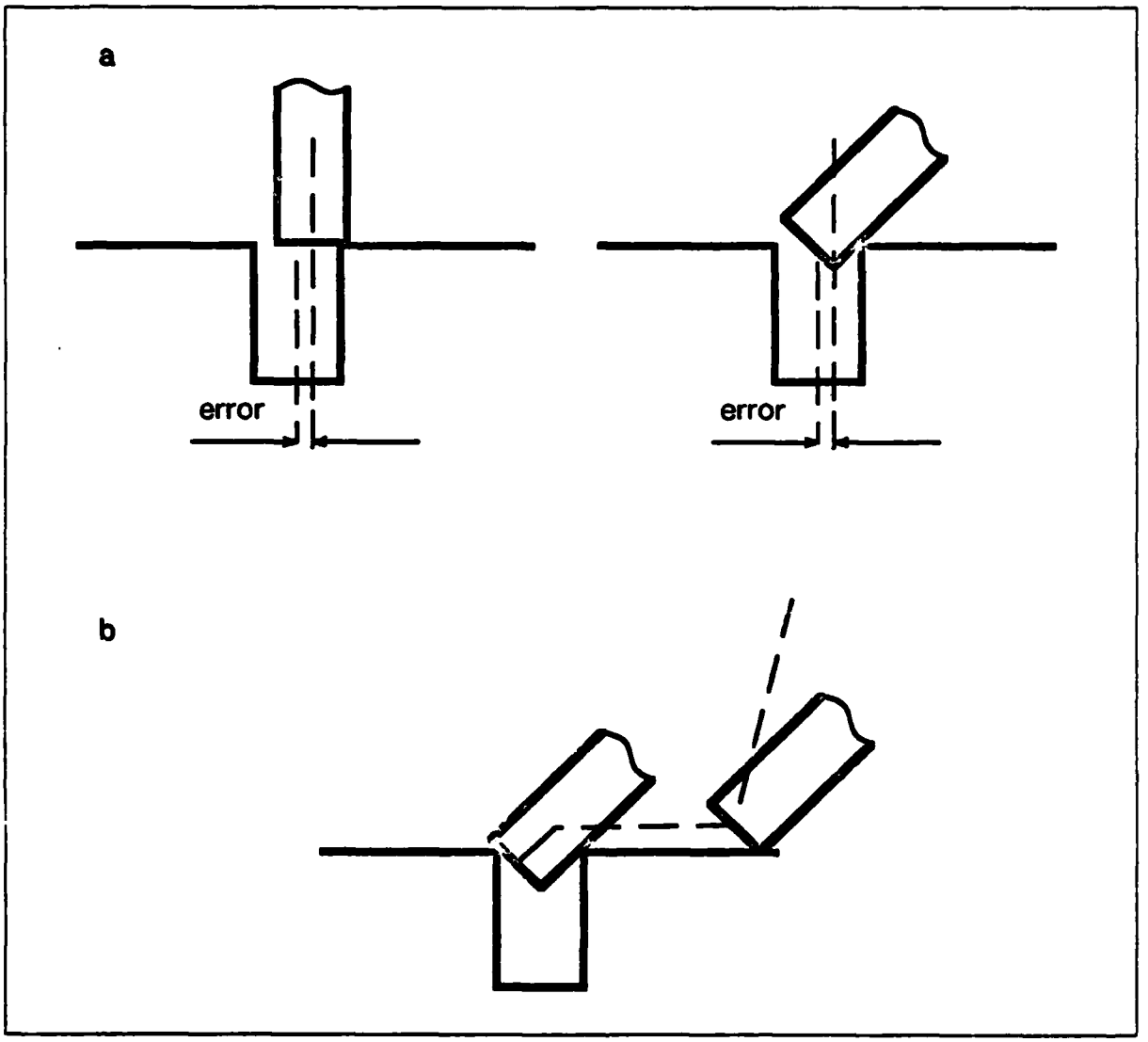


Figure 1. Two fine-motion strategies. (a) illustrates the value of tilting a peg before attempting to insert it in a hole. A comparable error causes failure without tilting, and success with tilting. (b) illustrates the use of the biased search. The peg is deliberately brought down to one side of the hole. This reduces or eliminates the chance of hitting the hole directly, but improves the chance of sliding into the hole on the second motion, because the direction to the hole is known.

There is one important difficulty in decomposing the sequence-planning problem this way: since we cannot guess the correct final motion, we will have to consider all possible final motions. We will use a type of motion which tends to move in a particular direction θ, but will slide along the surface of any object it strikes (see section I.A for details). Thus, the final motion can be any one of s_θ, for $\theta \in [0, 360)$. For each possible final motion s_θ, we define a *convergence region* R_θ to include every position c_{init} such that if motion s_θ is applied with the robot initially at c_{init} the goal G will be attained. In general,

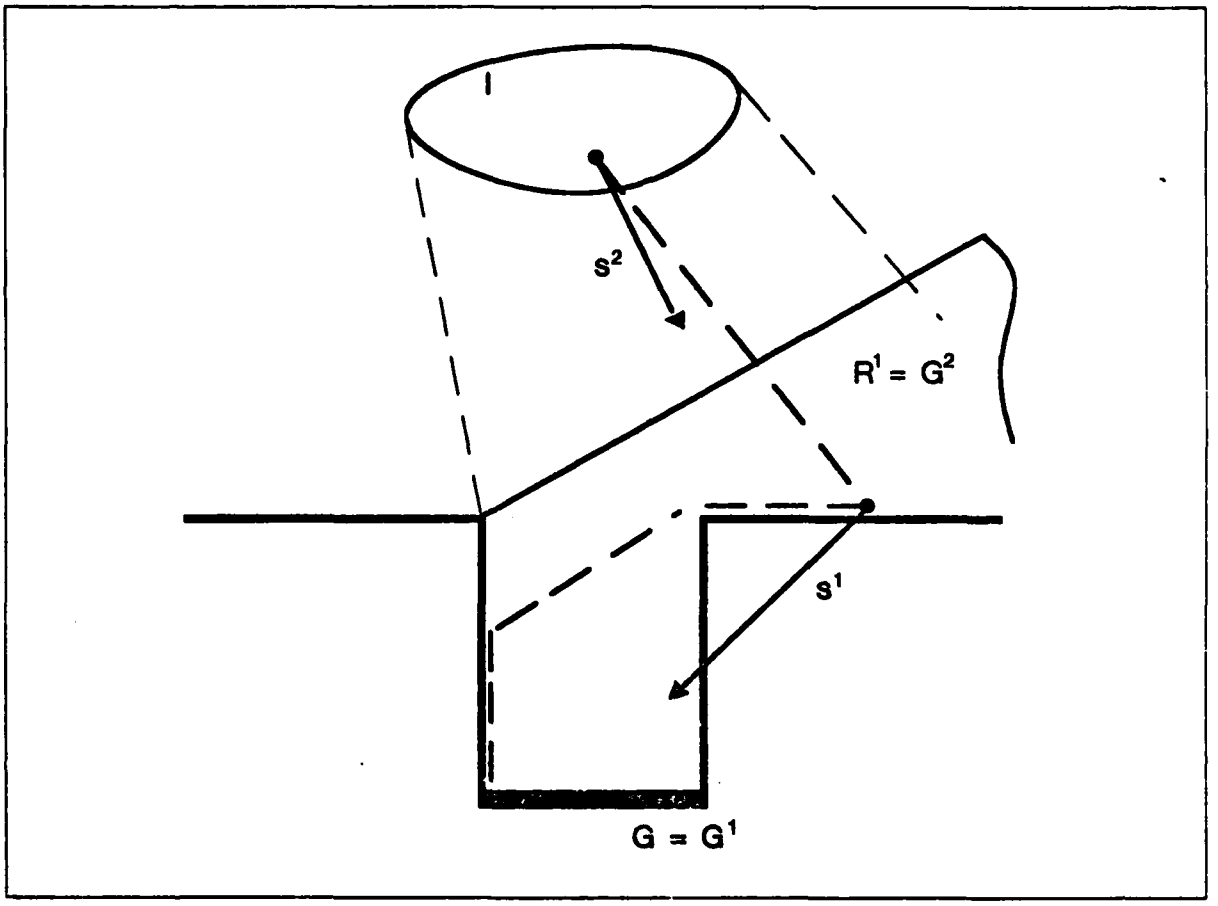


Figure 2. Example of planning a motion strategy. Hypothesizing single motions allows construction of subgoals. A chain of subgoals is constructed by iteration, which may ultimately include the input set I. In this example, a final command motion s^1 is hypothesized, which will succeed from anywhere inside the convergence region R^1. R^1 is then taken as the subgoal G^2, and command motion s^2 is considered. The convergence region R^2 contains the entire plane. Since this includes the initial set I, the planner concludes that the program consisting of s^2 followed by s^1 will attain the goal G from anywhere in I.

there can be many different convergence regions for goal G and motion s_θ (see figure 8c, for example), but in this example we will construct just one convergence region for each motion considered.

Consider the peg-in-hole problem, and assume that the control error may give a directional error as great as 15 degrees, and that friction will cause the robot to stick on a surface if its path is within 15 degrees of the surface normal (see section I.A for justification of this assumption). There are six types of convergence regions R_θ, which are illustrated in figure 3. If $\theta \in [-15, 195]$, then R_θ is the goal G itself, where the index θ indicates the angle with respect to the horizontal. These motions would tend to move the robot away from the hole, and will succeed only if the robot begins at the goal. (This assumes the motion may be halted before it begins.)

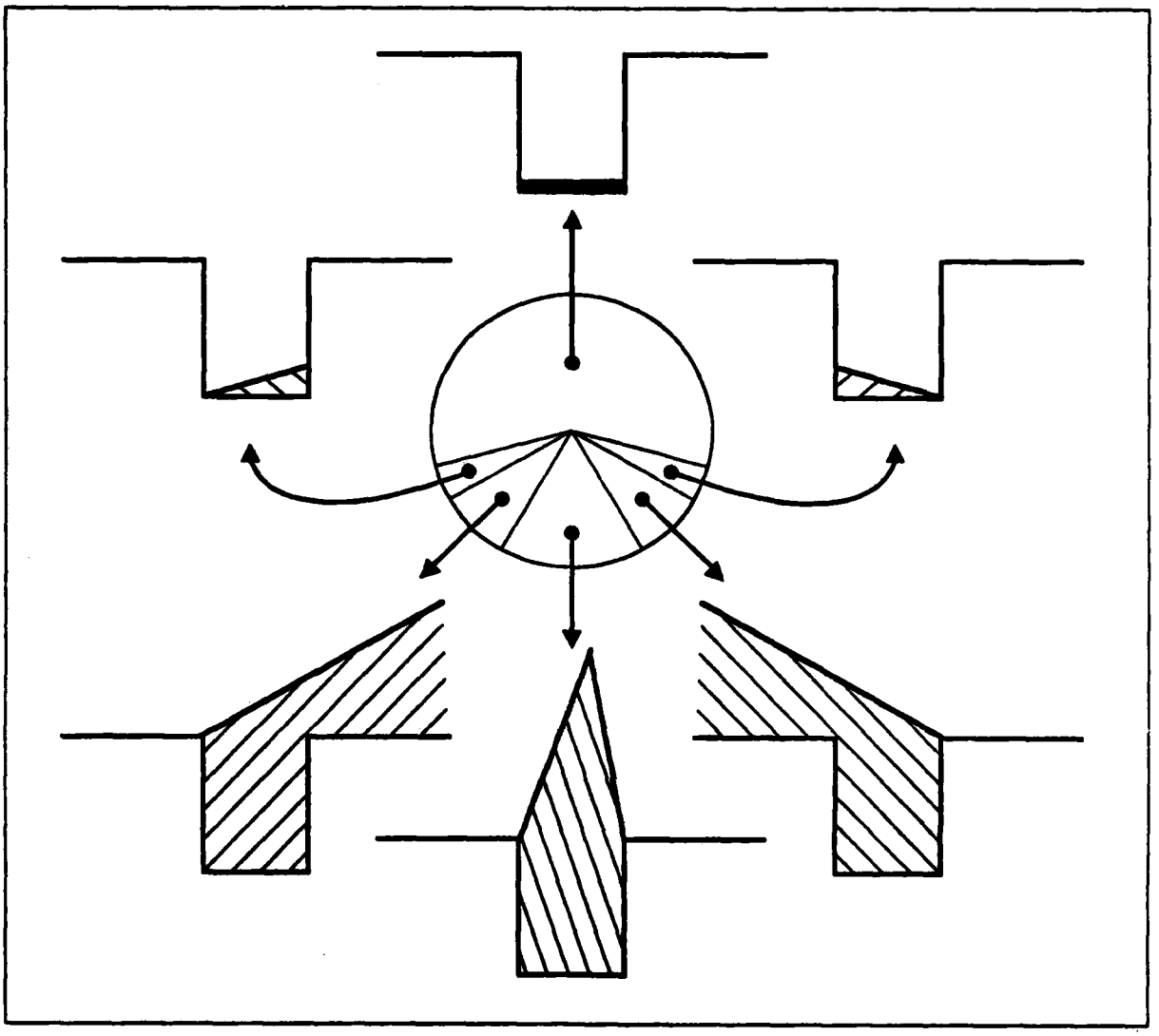


Figure 3. Convergence region types for the bottom of the hole. Although there are an infinite number of different convergence regions, each is a simple variation of one the convergence regions illustrated here. Each sector of the circle includes values of θ, the commanded motion direction, which would give rise to a convergence region of the type illustrated.

If $\theta \in (195, 210]$, then R_θ is a small wedge-shaped region lying at the bottom of the hole, bounded above by a line making an angle of $\theta - 15$ degrees.

If $\theta \in (210, 240)$, then R_θ is a large wedge-shaped region including the hole and the surface to the right of the hole. The reason such a region is so large is that if the robot strikes the horizontal surface anywhere to the right of the hole, it will slide leftwards into the hole. If it strikes the vertical wall of the hole, it will slide to the bottom of the hole. The convergence region is bounded above by a line making an angle $\theta - 15$ degrees. If the robot is initially above this line, it is possible for the robot to miss the hole to the left,

assuming the worst possible control error.

If $\theta \in [240, 300]$, then R_θ includes the hole and an angular region above the hole. These regions are much smaller than the wedge-shaped regions considered in the previous paragraph. This is because if the robot were to strike the horizontal surface at such a large angle, it would get stuck due to frictional forces (see section I.A). The region is bounded on the left by a line making an angle $\theta - 15$ degrees, and on the right by a line making an angle $\theta + 15$ degrees. These two lines intersect above the hole, giving R_θ its angular shape. If the controller were perfect, the region would be an infinite band.

The other two types of convergence regions, obtained for intervals $(300, 330)$ and $[330, 345)$, are left-handed versions of the two wedge shapes we have already considered. We have thus analyzed all possible final motions.

In planning a sequence of motions, we first determine whether the initial set I is a subset of any convergence region R_θ. If it is, then the single motion will suffice to solve the problem. If a single motion is not sufficient, as in our example, then we must repeat the above analysis for the next-to-final motion, with the goal of attaining one of the R_θ's computed previously. This time there are motions whose convergence regions include I. For example, the motion s^2 in figure 2 will reliably carry I to the wedge-shaped region R^1. Thus we have derived a two-motion sequence which solves the planar point-in-hole problem.

The example above illustrates the basic approach. We begin with a method which computes convergence regions for single motions, and executes a motion if one of the convergence regions is attained. Using this capability as the kernel, we obtain sequences of motions by simple backward-chaining. However, details remain which are handled in sections I.A and II.

A. Elements of the approach.

The example above involved the motion of a point in a plane with simple obstacles. In this section we will discuss models—the means by which the approach may address real manipulation problems. The model uses five fundamental elements: configuration space; guarded moves; generalized dampers; Coulomb friction; and uncertainty. These elements, taken together, provide the combination of simplicity and fidelity that we need for automatic planning.

Configuration space.

The point-in-hole problem is simple partly because the shape of the "robot" is simple. Point robots do not occur in nature; it is necessary to reason about the motions of complex linkages of solid objects. The best conceptual tool for addressing this problem is *configuration space*. Define a *configuration* of a system to be the parameters required to completely

determine the position of every point in the system. For a six degree-of-freedom robot, six parameters are required; in general, we will assume that n parameters are required. Take the n-tuple of values specifying a configuration to be a point in n-dimensional space: configuration space. The final step is to represent obstacles in configuration space. Again this is conceptually simple: we represent an obstacle in configuration space by the set of points (robot configurations) which would cause interference between the robot and the obstacle.

A simple example of configuration-space is illustrated in figure 4. This shows the configuration-space transformation for an upright peg insertion and for a tilted peg insertion. The value of the peg-tilting strategy is readily apparent in configuration space—a broad notch makes a much better target than a narrow slot.

The importance of this tool cannot be overstated: it transforms a seemingly impossible problem—reasoning about the motion of a collection of three-dimensional volumes—to a merely very difficult problem—reasoning about the motion of a (n-dimensional) point. For the application of configuration space in motion planning and compliance, see [Lozano-Pérez 1981, Mason 1981, Lozano-Pérez 1983, Brooks and Lozano-Pérez 1983].

Guarded moves.

The *guarded move* [Will and Grossman 1975] is a very good tool for making the robot adapt to variations in the task. The basic idea is to execute a motion while carefully monitoring the sensors. The motion is terminated when a pre-specified sensory event occurs—usually when a collision is detected. The effect is an important one: the manipulator may be positioned accurately, relative to some geometrical feature, even if the location of the feature is uncertain. All the motions we will consider are guarded moves, and the derivation of the terminating sensory event is discussed in detail in section II.A.

Generalized dampers.

The generalized damper [Whitney 1976] is a simple device for obtaining compliant motion. The name derives from the similarity with a mechanical viscous damper, or dashpot, which maintains a proportional relationship between velocity and force. The generalized damper is defined by the equation:

$$f = B(v - s)$$

where f is a vector of external generalized forces, v is a vector of generalized velocities, s is the *nominal velocity*, and B is an n-by-n matrix of damping factors. For a manipulator, f and v could be expressed in joint space or in "Cartesian space." For problems involving the motion of a point in a plane, v refers simply to the two components of velocity and f refers to the two components of force. The nominal velocity s is the control input. In the absence of contact forces the actual velocity will agree with the nominal velocity. When

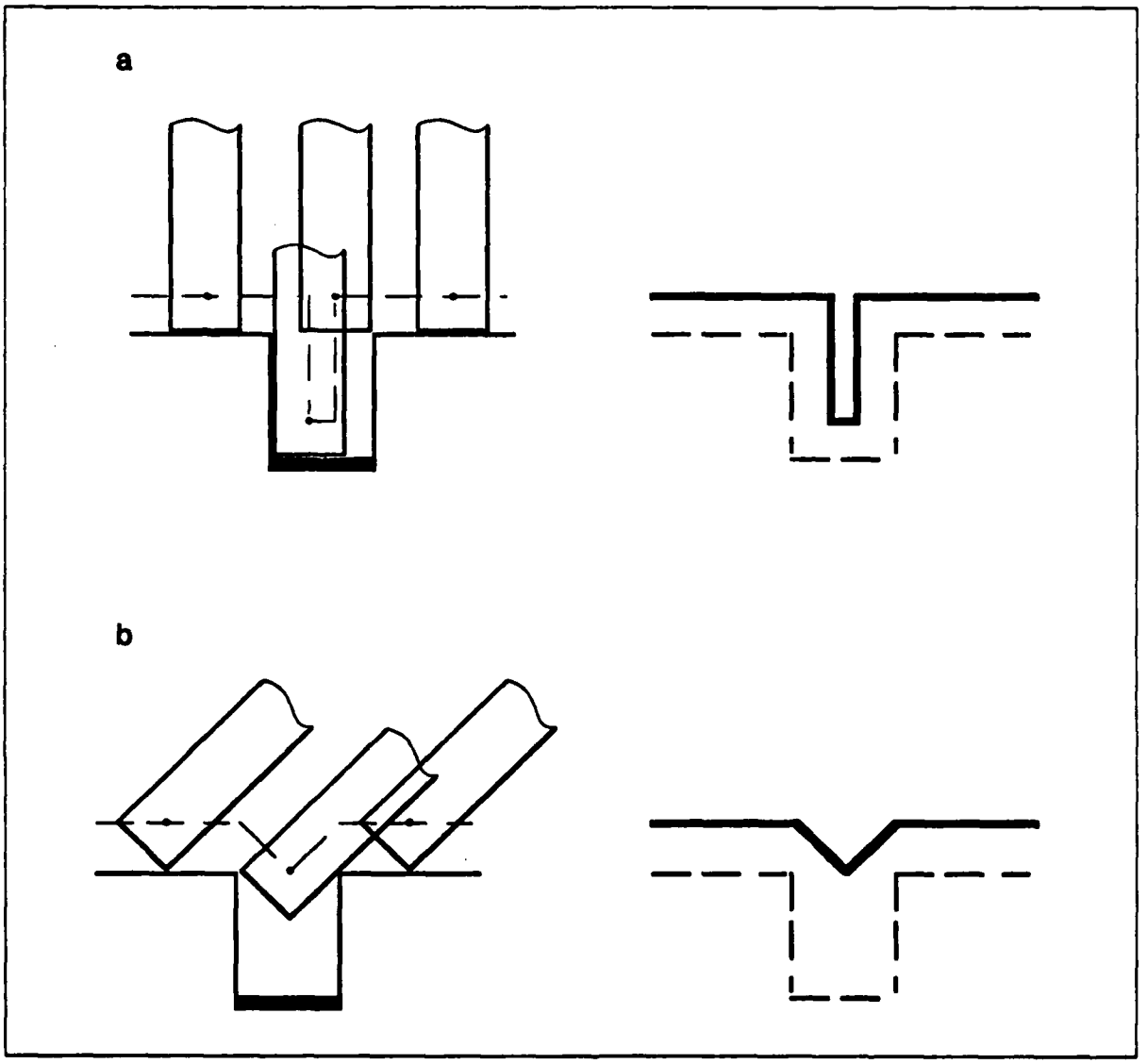


Figure 4. Configuration spaces for peg insertions, allowing translational motions only. (a) shows an upright peg insertion; (b) a tilted peg insertion. The original problem appears on the left, and the corresponding configuration-space surfaces are shown on the right.

external forces are applied, the velocity will depart from the nominal velocity in a linear fashion.

Many damping matrices B are possible, and some produce interesting (and bizarre) behavior. However, for our purposes we will assume the most mundane of damping matrices, $B = bI$. Thus the robot will adjust its velocity away from any perceived force. For a discussion of the generalized damper and other methods of compliant motion, see [Mason 1983].

Modelling contact.

To model the behavior of the generalized damper when contact occurs, a model of frictional force is required. Fortunately, Coulomb friction is both simple and fairly realistic. Let μ be the coefficient of friction, and let α be $\tan^{-1}\mu$. We will assume the static and dynamic coefficients of friction are equal. For two- and three-dimensional translation, if the nominal velocity s makes an angle of less than α with the surface normal, the robot will stick when it hits the surface. If the angle is greater than α, the robot will slide along the surface. The actual path of the robot will be the normal projection of the nominal path, although the time dependence is a little more complicated. We could hardly ask for simpler behavior—the robot follows a simple sequence of straight lines. [Erdmann 1983] obtains similar results for problems involving rotations and moments. A more rigorous discussion of sliding with Coulomb friction and generalized damping appears as an appendix in [Lozano-Pérez, Mason, and Taylor 1983].

B. Discussion.

Previous work.

The approach explored in this paper was first described in [Lozano-Pérez, Mason, and Taylor 1983]. That paper included a formal description of the stateless-predicate variation and an informal, but more detailed, description of a variation using backtracking. Both variations are explored in section II.B. For a discussion of other related work in fine motion, compliance, and automatic planning, see [Lozano-Pérez, Mason, and Taylor 1983].

Implications.

The main goal of this paper is a theoretical exploration of the scope of an approach to automatic robot programming. This goal has led to two important results: (1) construction of a formal description of the approach (FMP); (2) proofs of correctness and bounded-completeness of FMP.

The pursuit of a formal description has clarified a number of difficulties with the approach, and with general robotic manipulation. FMP provides precise answers to such questions as "how much information should be saved during a motion program?" "how do we interpret contact sensors?" "how do we combine information from models, controls, and sensors?" and "how is uncertainty anticipated in the construction of subgoals?" Although FMP necessarily addresses these questions in a limited context, the answers provide insights into general manipulation problems.

The proofs of correctness and bounded-completeness validate the design of FMP, and define the theoretical scope of the approach. In the process of designing and implementing planning procedures, choices can be made with a full appreciation of how the scope will be affected. If we can implement FMP, then we can be certain of the ability to obtain solutions to an important class of manipulation problems. When compromises are necessary to

reduce the technique to practice, they can be made knowledgably. Ultimately, we may have a number of approaches for manipulation and manipulation planning. The choice of which to apply to a given manipulation problem could be guided by the theoretically-determined scope of each approach.

One point that must be emphasized is that FMP has not been implemented. Despite the fact that the description has the form of a procedure, we must resist calling it an algorithm—it is not detailed enough to be considered an effective procedure. Some of the sets which must be computed do not even have constructive definitions, much less a method of representing and computing them. It is even possible that they are not computable.

The bounded-completeness of FMP shows that if a solution of the requisite form exists, FMP will find it. It does not address the question of whether solutions exist—of whether a bounded sequence of straight-line, generalized-damper, guarded motions can perform useful manipulation. This is an important area for further work, both theoretical and experimental.

II. Synthesis

This section presents detailed descriptions of the synthesis procedure FMP and four variations. We will address the issue of completeness for each of the variations. Correctness and bounded-completeness of FMP is taken up in section III. A summary of the results is presented in table 1.

Variation	Complete?	Performance
FMP	yes	best worst-case
Backtracking	no	
Stateless predicate	?	worse than FMP
Subgoals per motion	yes	same as FMP
Look-ahead predicate	yes	better than FMP

Table 1

The table indicates whether each variation is bounded-complete. The table also lists the performance of the motion programs constructed by each approach, compared against the solutions found by FMP. The *backtracking* variation, which was investigated because it offers advantages in implementation, is not complete. There are problems which this variation cannot solve, even though bounded solutions exist. The *stateless predicate* variation is an early formulation of FMP described in [Lozano-Pérez, Mason, and Taylor 1983]. We do not know if it is complete, but we do know that its solutions are sometimes inferior to those constructed by FMP. The *subgoals per motion* variation offers advantages in implementation, and is conceptually simpler than FMP. In fact, the description in section I corresponds to the subgoals-per-motion variation. The solutions found by this variation are identical to those found by FMP. Finally, the *look-ahead predicate* variation is a simple variation whose solutions are never worse, and sometimes better, than those constructed by FMP. Detailed descriptions of these variations, and derivation of the results summarized in table 1, follow the description of FMP.

A. FMP (Fine-Motion Planner).

Before the formal description of FMP is presented, it is necessary to consider in more detail the termination predicate, which terminates a motion when it detects that a goal or subgoal has been achieved, and the construction of subgoals, which is fundamental to the entire approach.

Notation

First, it is necessary to survey some of the notation required. We use s to indicate a

nominal velocity, and R to indicate convergence regions, c and v to refer to configuration and velocity, respectively, and c_{init} to refer to the configuration at the beginning of a motion.

When it is necessary to distinguish between different motions in a sequence, we use a superscript indicating the reverse order of the motions. Thus s^1 refers to the final nominal velocity, R^1 refers to the corresponding convergence region, G^1 refers to the corresponding goal, etc.

We assume position and velocity sensors—force measurements can be transformed to velocity measurements using the damping equation, and velocity is easier to work with. Our model of error associates a tolerance with each sensory and control variable: ϵ_c, ϵ_v, and ϵ_s refer to tolerances associated with position sensing, velocity sensing, and control, respectively. We use an asterisk to refer to measured or commanded, rather than actual, values. Thus c^* is the position sensor value, v^* the velocity sensor value, and s^* is the commanded nominal velocity. $B(x)$ is the set of all values within ϵ of x; thus, for example, $B(c^*)$ is the set of all configurations c consistent with the sensor value c^*.

Termination predicate.

The termination predicate continuously monitors the sensors during a motion, and terminates the motion as soon as it detects that a subgoal has been attained. Although the last motion need worry only about a single goal, the other motions will have a multitude of subgoals to consider—one for each convergence region constructed by the previous planning stage. The termination predicate must watch for the attainment of any subgoal, and then must return the identity of the subgoal attained. We can imagine that the termination predicate is given a large directory of subgoals, with one subgoal on each page of the directory. The job of the termination predicate is to wait until it is certain that one of the subgoals has been achieved, whereupon it rips the corresponding page from the subgoal directory and returns it.

How can the termination predicate decide that a subgoal has been achieved? The success of the approach depends on the ability of the termination predicate to detect a subgoal as soon as possible. The information available to the termination predicate includes: (1) R, the set of all possible locations of the robot at the beginning of this motion; (2) s^*, the commanded nominal velocity; (3) c^* and v^*, the position sensor data and velocity sensor data, respectively, which are functions of time; and (4) t, the time. It also has access to the geometrical model of the environment. When the termination predicate is first constructed, a directory of trajectories is compiled, which includes an entry for every robot trajectory consistent with R and s^*. Given the initial position c_{init}, the actual nominal velocity s, and the geometrical model, the trajectory is completely determined, so the trajectory directory includes an entry for every pair (c_{init}, s) with $c_{init} \in R$ and $s \in B(s^*)$.

Now we can define the actions of the termination predicate during the motion. At every time t, the predicate examines every page of the trajectory directory. On each page is a trajectory which represents a prediction of (c, v), the position and velocity of the robot. If this prediction is consistent with the current sensor values (c^*, v^*), i.e. if c is in $B(c^*)$ and v is in $B(v^*)$, then all is well, and the next page is examined. However, if the trajectory is not consistent with the sensory data, then the prediction is invalid; in this case, the termination predicate rips out and discards the page. As this operation is applied to each page of the trajectory directory, the validity of the trajectory directory is preserved. When the procedure is complete the trajectory directory comprises exactly those trajectories which are consistent with model, control, and sensory data, both present and past.

After the task of updating the trajectory directory is complete, the termination predicate constructs a set Q including the current position of every trajectory in the trajectory directory. This gives the set of all possible locations, i.e. all locations consistent with model, control, and sensory data. It is a simple matter to leaf through the subgoal directory, looking for Q. If it does appear in the subgoal directory, the termination predicate halts the motion and returns Q. This set will eventually be passed to the termination predicate of the next motion as the set R.

It may seem strange that the predicate looks for a subgoal equal to the set of all possible positions Q, rather than for a subgoal including Q. In the next section we will define the set of subgoals to include *every* set from which a *single motion* will be able to attain the subgoal, even sets which are subsets of other subgoal sets. It should be obvious, then, that if G_α is a subgoal, every subset of G_α is also a subgoal. Thus if Q is a subset of some subgoal, it will appear in the directory as a subgoal in its own right.

Note that this procedure satisfies the specified criterion of returning the identity of the subgoal as soon as possible. If the termination predicate has not halted the motion, it is because for every subgoal there are locations of the robot which are consistent with all available information but which are not in the subgoal.

Construction of subgoals.

The proper definition of subgoals (convergence regions) presents the greatest difficulty in the definition of FMP. Let $\{G_\alpha\}$ be the set of goals, and let $\{R_\beta\}$ be the set of all subgoals, which is to be constructed and passed to a recursive call. A set should be included in $\{R_\beta\}$ if, and only if, attaining that set will allow the planner to attain one of the $\{G_\alpha\}$ in a single motion. To define subgoals in more detail, imagine that the recursive call has returned one of the subgoals R, which is now guaranteed to include the true location of the robot. Let $S(c^*_{init}, R, \{G_\alpha\})$ be the set of all command nominal velocities s^* such that the termination predicate, constructed as described above, is certain to terminate. Now, let $P_R(\{G_\alpha\})$ be the set of initial locations c_{init}, such that for all possible c^*_{init}, $S(c^*_{init}, R, \{G_\alpha\})$ is certain

not to be empty. Then there is a good nominal velocity if and only if the robot's true initial location c_{init} is in $P_R(\{G_\alpha\})$. R is the set of all possible c_{init}, so there is good nominal velocity if and only if $R \subseteq P_R(\{G_\alpha\})$. Let us refine the definition of P_R slightly and state it formally:

$$P_R(\{G_\alpha\}) = \{c_{init} \in R \mid \forall c^*_{init} \in B(c_{init}), S(c^*_{init}, R, \{G_\alpha\}) \neq \emptyset\}.$$

We do not include points outside R, simply because it would be silly to worry about them—such a point would be a good place for the robot to be, provided that it is someplace else. With this refinement, we can restate the criterion for suitable subgoals: R is a suitable subgoal if and only if R satisfies the equation $R = P_R(\{G_\alpha\})$. As noted above, if R is a suitable subgoal, so is every subset of R.

The difficulty is that $P_R(\{G_\alpha\})$ depends on R, as indicated by the subscript. The equation $R = P_R(\{G_\alpha\})$ tells us, in principle, how to test for suitable subgoals, but it does not suggest a means of constructing suitable subgoals. In section II.B, we will see some improvement, but this remains the biggest impediment to further elaboration of the approach.

Formal description: nomenclature.

c	configuration.
c_{init}	configuration at beginning of a motion.
v	velocity.
s	nominal velocity.
c^*	observed configuration.
c^*_{init}	observed configuration at beginning of a motion.
v^*	observed velocity.
s^*	commanded nominal velocity.
t	time.
C	configuration-space, i.e., the set of all configurations.
$B(c)$	the "uncertainty ball" of configurations; i.e., the set of all configurations whose distance from c is within the tolerance of the position sensor.
$B(v)$	the "uncertainty ball" of velocities.
$B(s)$	the "uncertainty ball" of nominal velocities.
$\{G_\alpha\}$	current goal set. We wish to move the robot to one of the goals and return the identity of the goal.

$p(c^*, v^*, t)$ the termination predicate.

$S(c^*_{init}, R, \{G_\alpha\})$ is the set $\{s^* \mid p \text{ terminates}\}$. By construction of the predicates, guaranteed termination implies guaranteed attainment of a goal. So for a given observed initial configuration and accomplished subgoal R, this gives the set of all winning strategies, where a strategy comprises a command nominal velocity and a termination predicate.

$P_R(\{G_\alpha\})$ is the pre-image $\{c \in R \mid \forall\, c^*_{init} \in B(c), S(c^*_{init}, R, \{G_\alpha\}) \neq \emptyset\}$.

$\{R_\beta\}$ The sets of configurations R such that the pre-image $P_R(\{G_\alpha\})$ includes all of R, i.e., $P_R(\{G_\alpha\}) = R$. This is the subgoal set—attaining an element of this set by a recursive call will allow us to satisfy the current goal set.

R the subgoal attained by recursive call to the planner, hence the set of all possible robot locations at the beginning of the motion.

`MotorCommand(`s^*`)` execution of this program statement transmits the commanded nominal velocity to the controller, causing the manipulator to execute the planned generalized damper strategy.

Formal description: procedure.

```
Procedure FMP(I,{Gα})

        Compute {Rβ}
        If I is in {Rβ}
                Then R <- I
                Else R <- FMP(I,{Rβ})
        s* <- choose(S(c*init, R, {Gα}))
        Compile p
        t <- 0
        MotorCommand(s*)
        L       If p(c*,v*,t) signals termination Then Return(Gα)
                Increment t
                Go L

        End FMP
```

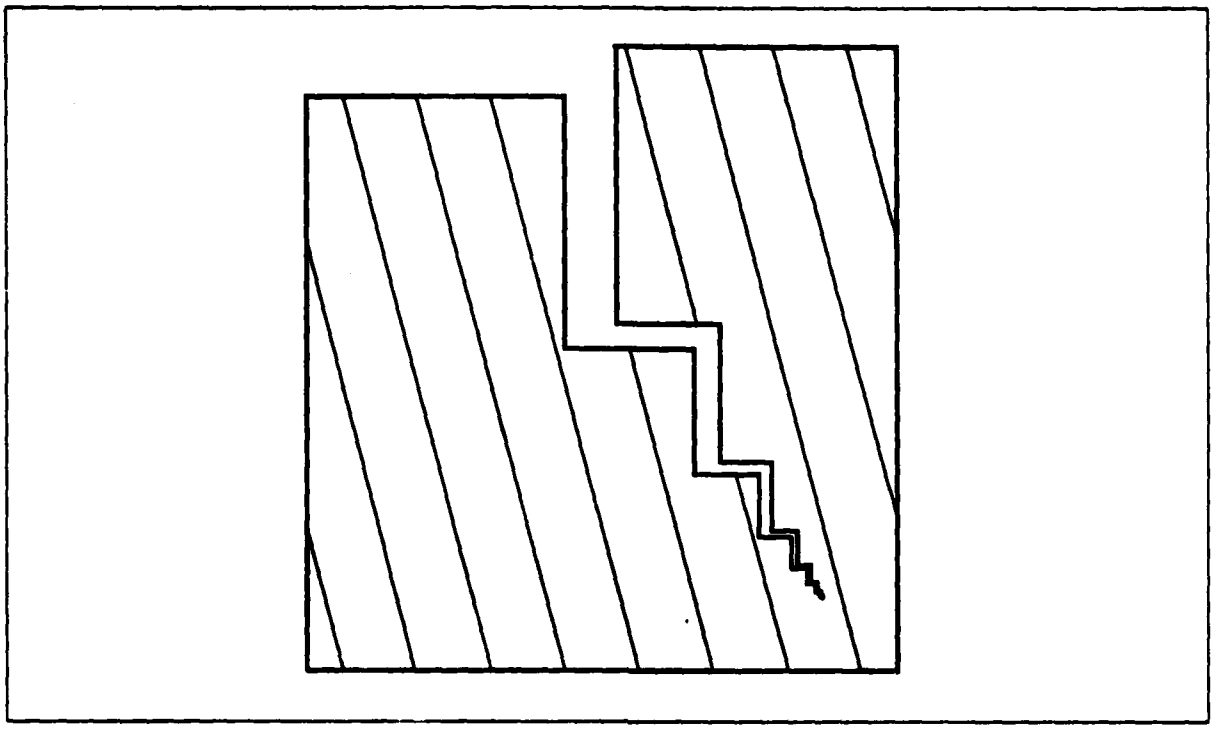


Figure 5. A problem requiring an infinite number of steps (assuming a large enough coefficient of friction). Assuming constant velocity and zero motion switching delay, the robot can reach the end of this infinite staircase in finite time.

Examples; discussion

There are a number of aspects of FMP which merit discussion. First, we refer to FMP as a planner, but it also executes the plan. The recursive back-chaining is the planning stage. If the backward-chaining terminates, i.e. if $I \in \{R_\beta^N\}$ for some N, then we say that the planner has converged, and we can think of the chain of subgoals as a plan. Execution of the plan occurs as the recursion unwinds. This structure implies one obvious restriction on the form of the plans that can be found: there must be a bound on the number of motions required to achieve the goal. Figures 5 and 6 are examples of manipulation problems which have solutions, but which do not have solutions of bounded-length.

Figure 7 shows two subgoals R_α^1 constructed for the peg-in-hole. These sets do not conform to any of the types described in the introductory example (figure 3), and require some explanation. Figure 7a contains the familiar cone and portions of the large wedges. If it is given that the robot is in the union of these three components, the question is whether a the planner can attain the goal in a single motion. The answer is yes; to plan the single motion, we first examine the position sensor. Since the three elements are never closer than $2\epsilon_c$, the sensor value can be within ϵ_c of just one of them. Thus we can determine which of the three components actually contains the robot, and plan the motion accordingly.

Figure 7b is similar. Given that the robot is in the shaded region, we plan a motion

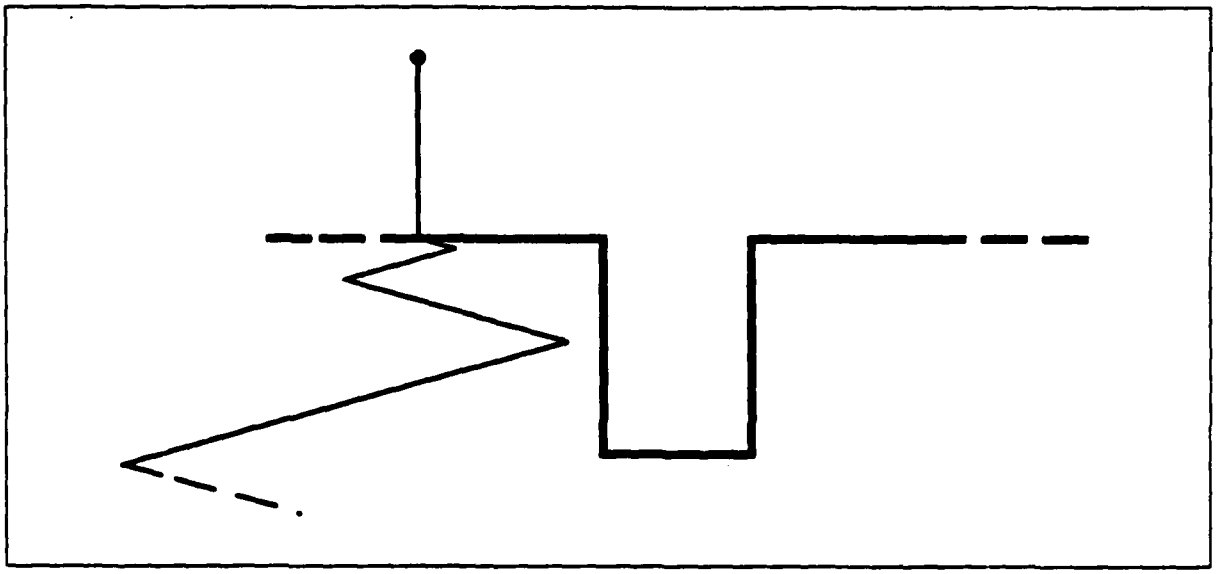


Figure 6. A problem requiring a finite, but unbounded, number of steps; $\epsilon_c = \infty$, $\epsilon_v = \epsilon_\theta = 0$, $I = C$. The robot has no position feedback, and has no idea whether to search to the left or the right for the hole. In order to find the hole, the robot must sweep back and forth, in an ever-enlarging search. Such a plan must succeed eventually, but there is no upper bound on the number of motions.

as follows: if the position sensor value is right of center, use a left-and-downward motion, else use a right-and-downward motion. Since a single motion attaining the goal can be determined, the shaded region does constitute a valid subgoal. The important property in this case is that the two wedge components are separated by $2\epsilon_c$. If the robot is in the middle region, it does not matter whether the left or right motion is applied.

Although these subgoals are more complicated (and confusing) than the subgoals illustrated in figure 3, there is no need for anguish. In the next section, "Subgoals per motion," a variation on FMP is described which has performance equivalent to FMP, but which uses the simpler subgoals of figure 3.

B. Variations.

The form of FMP reflects the goal of formally exploring the capabilities of the approach. Other goals, implementation in particular, have led to other formulations, whose capabilities we can now judge against the standard set by FMP.

Backtracking.

At each level, FMP constructs all suitable subgoals, and passes all of them to the next level. We have not yet devised means for representing and computing the set of all suitable subgoals. In the interest of demonstrating an actual implementation of the approach, we might consider a simpler problem: to construct single suitable subgoals. The subgoals could be constructed one at a time, and passed singly to the recursive call. Of course,

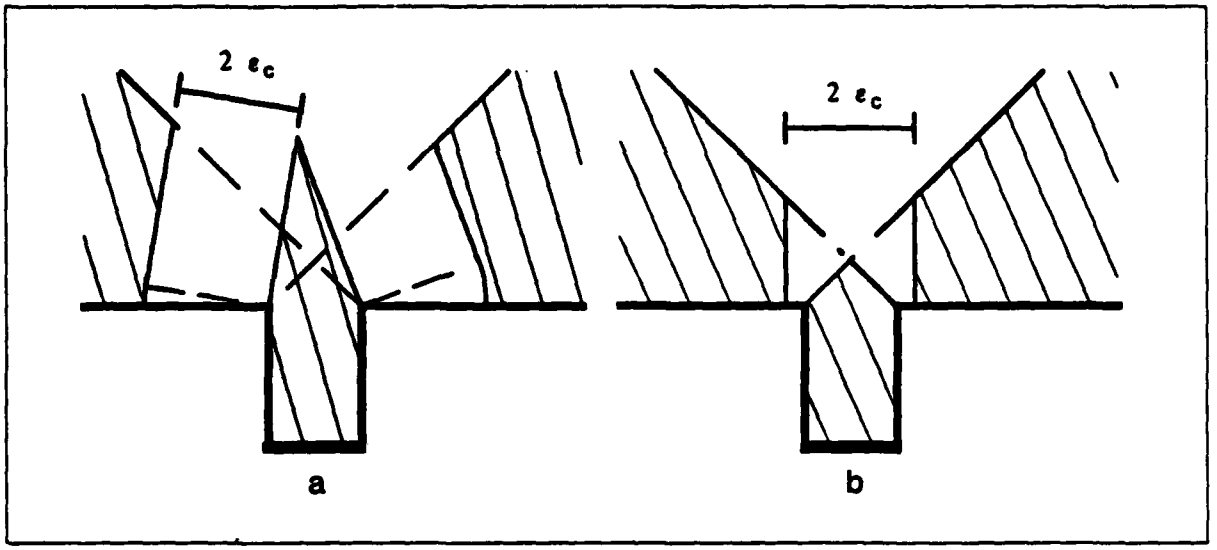


Figure 7. Two subgoals for the peg in hole. These subgoals may seem too big, but the planner can consult the sensors to further refine its idea of the robot's location.

if the planner chooses an unattainable subgoal, subsequent planning stages would fail, requiring that the planner backtrack, choose a different subgoal, and try again. The resulting procedure would have to search through the tree of subgoals.

The performance of this approach will obviously depend on the efficiency with which it can search the subgoal tree. However, no matter how efficient the search, there are certain problems which this approach cannot solve. Figure 8 is a problem (the point-on-hill problem) for which there is a simple two-motion solution, but which the backtracking planner cannot solve. The backtracking approach fails because it only allows the planner to consider one subgoal at a time. The planner is unable to use strategies of the form, "I don't know whether I'll attain G_A or G_B, but I'll know when I get there."

Stateless predicate.

The formal procedure FM described in [Lozano-Pérez, Mason, and Taylor 1983] differs from FMP in a seemingly innocuous way: the predicate only uses the current values of the sensors when deciding whether to terminate the motion. It doesn't rip the pages out of the trajectory directory permanently, it starts fresh for every iteration. Figure 9 shows a problem which takes FMP one motion, but which requires two motions by FM. With the downward nominal velocity, and assuming zero control error, there are two trajectories possible. At some intermediate time, the trajectories are separated by a distance greater than $2\epsilon_c$, so it is possible to determine which is the true trajectory. Unfortunately, by the time the robot is approaching the goal, the distance between the trajectories has closed, and FM cannot *remember* which of the two trajectories is the correct one. Without memory, the termination predicate must resort to a different approach using two motions. Thus, FM sacrifices some performance. I don't know yet whether it sacrifices bounded-completeness.

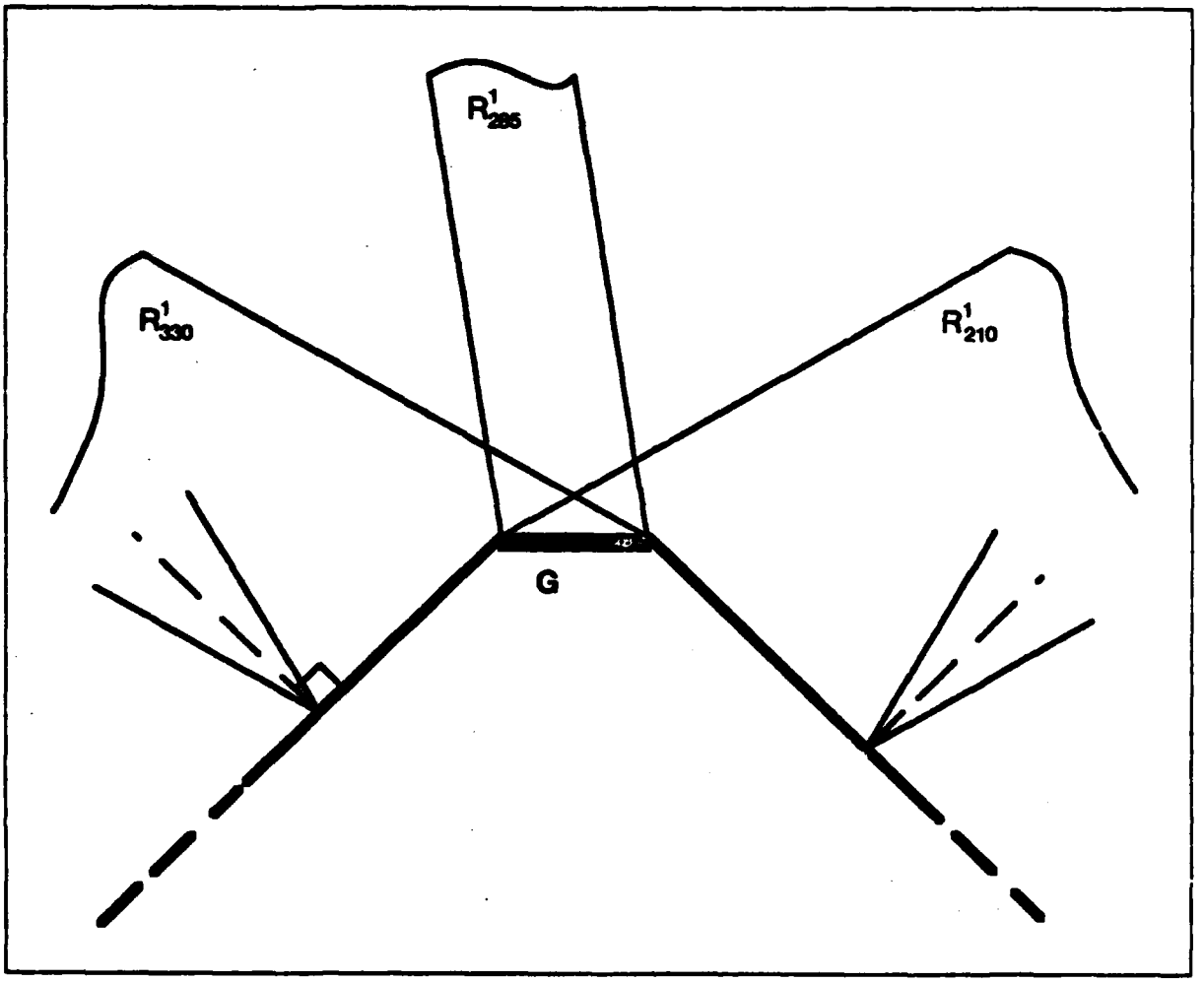


Figure 8a. The point-on-hill problem; $\epsilon_d = \infty$, $\epsilon_p = \epsilon_\theta = 0$, $I = C$. Since the robot has no position feedback, it must rely on contact with the hill. A simple strategy is to proceed downwards until contact. If the robot tends to the left at termination, it can move up to the right to attain the goal. Otherwise it should move up to the left. The backtracking variation cannot solve this problem; since it considers only one subgoal at a time, it cannot derive an either-or strategy. The best subgoal chain that the backtracking variation can construct is shown through three levels, after which no improvement is possible. (a) shows typical convergance regions at the first level. The best single subgoal is R^1_{210}, which is passed to the second level.

Subgoals per motion.

There is one important variation which, surprisingly, does not sacrifice bounded-completeness. It seems clear that we would like our subgoals to be as large as possible, to make the job of the recursive call as easy as possible. It would seem that the complicated, maximal, subgoals illustrated in figure 7 would be good ones. Fortunately, constructing such complicated subgoals is unnecessary. The simpler subgoals shown in figure 10 are sufficient to obtain bounded-completeness and to obtain the performance of FMP.

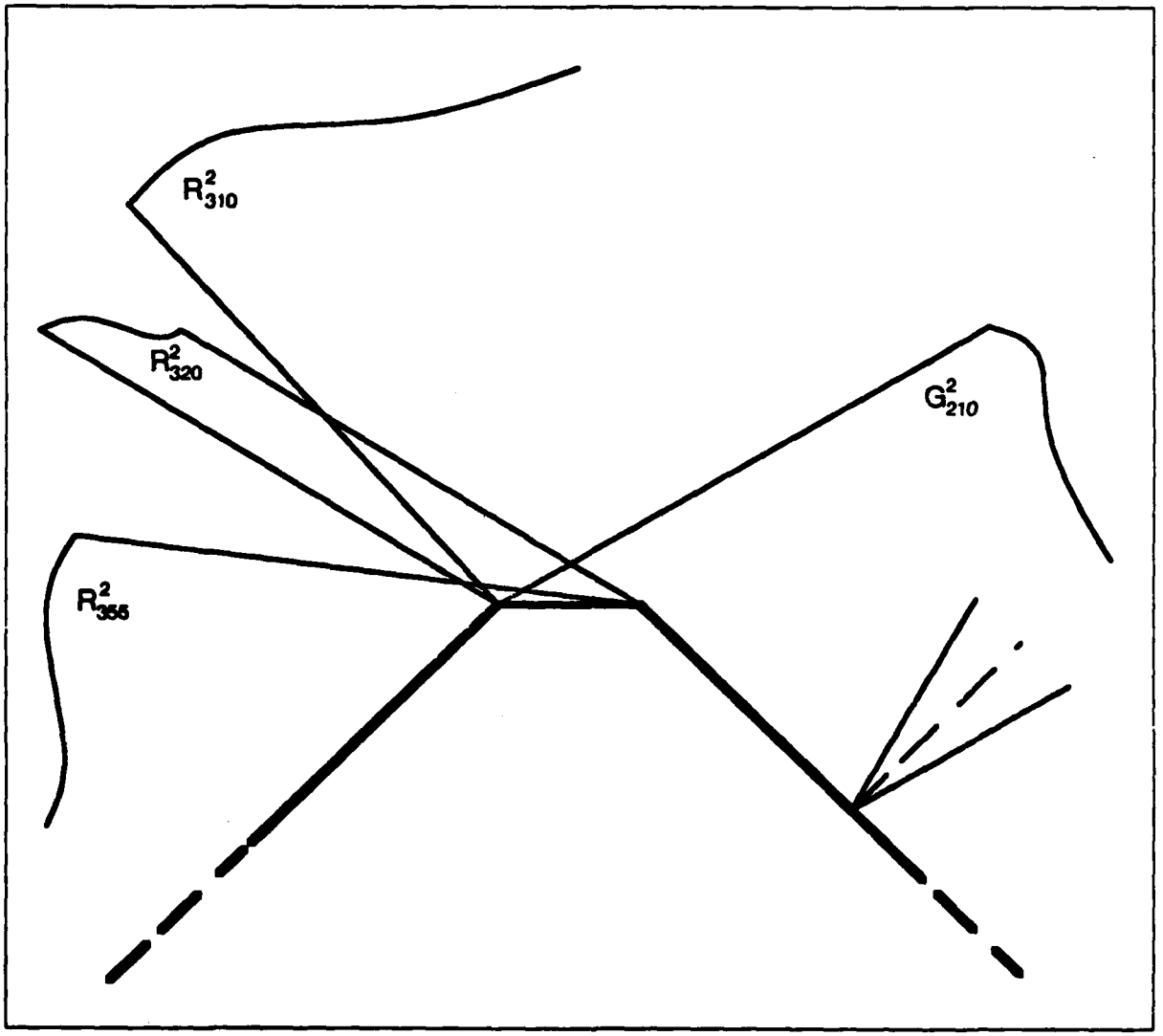


Figure 8b. The point-on-hill problem continued, showing the inability of the backtracking variation to solve the problem. (b) shows convergence regions constructed at the second level. The best is R^2_{310}, which is passed to the third level.

This result can best be explained by considering the use of the complex subgoal of figure 7. Let us suppose that the subgoal of figure 7b has been attained, and it is our job to plan the next motion. If the robot position c is in leftmost region, we require a right-and-downward motion; if c is in the rightmost region, we require a left-and-downward motion. There is no motion which works for both regions. In order to decide which motion to apply, we consult the position sensor. If c^* lies to the right of center, we know that c is either in the center region or the rightmost region, and the left-and-downward motion can be used. Otherwise the right-and-downward motion is applied. This complex set is indeed a proper subgoal—attainment of this set is sufficient to attain the goal in a single motion. Note, however, that all the information available when planning the next motion was also

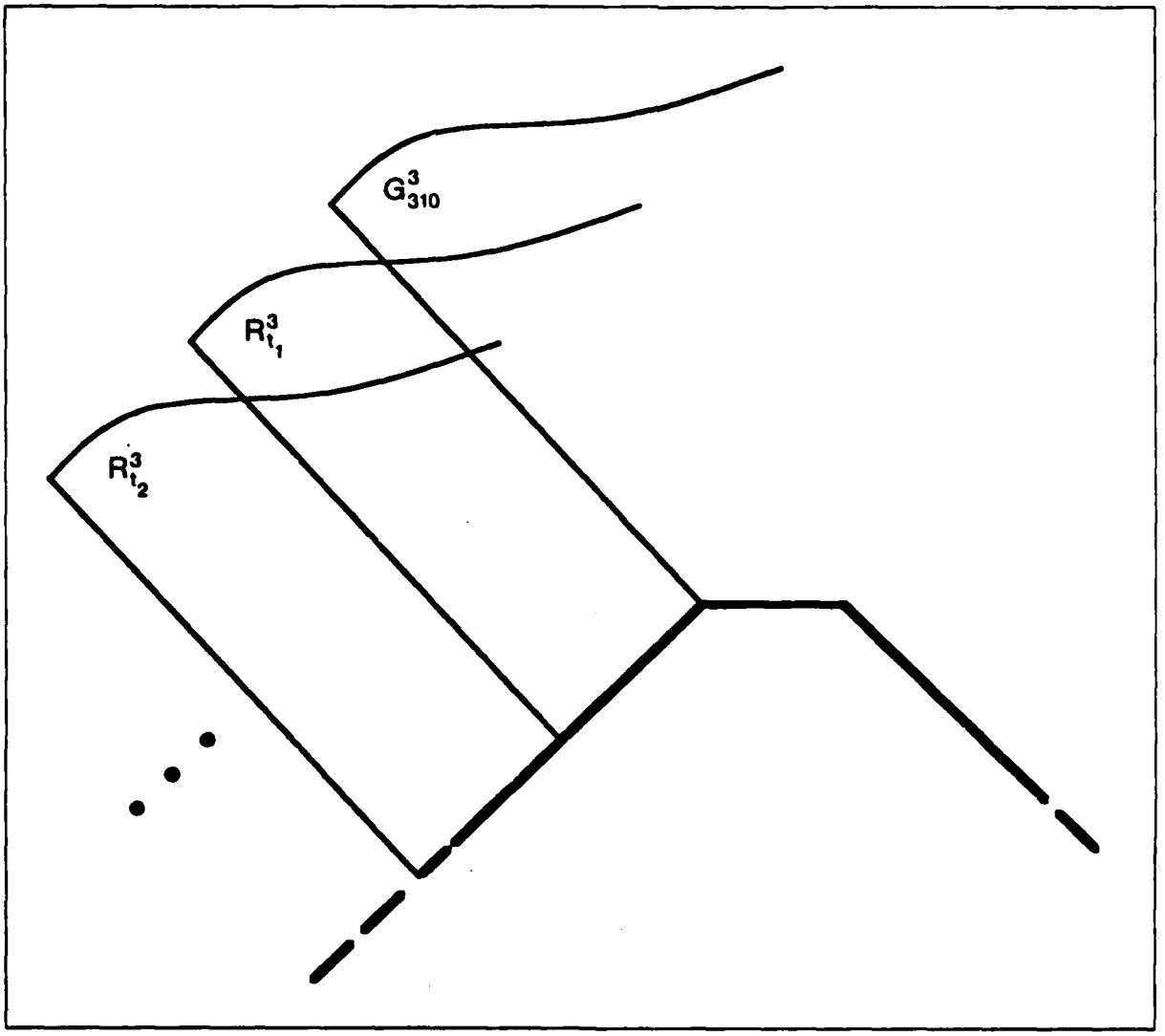


Figure 8c. The point-on-hill problem continued, showing the inability of the backtracking variation to solve the problem. Convergence regions constructed at the third level are shown here. These regions correspond to motions which use elapsed time to terminate. For example, given that the robot is in $R^3_{t_1}$, the planner can calculate the maximum possible distance to G^3_{310}. For a motion of known velocity in the appropriate direction, this gives a maximum time t_1 to arrival in G^3_{310}. Termination of the motion consists simply of waiting the calculated time before terminating. Although the convergence regions can be made arbitrarily large by considering arbitrarily large waiting times, no convergence region includes the entire plane. Thus, if the initial set I consists of the entire plane, the backtracking variation cannot find a solution.

available when the previous motion terminated. Thus, if we can figure out that c is either in the center region or the rightmost region, the termination predicate could have figured it out, too. In this case, passing the complex subgoal offers no advantage over passing the components independently.

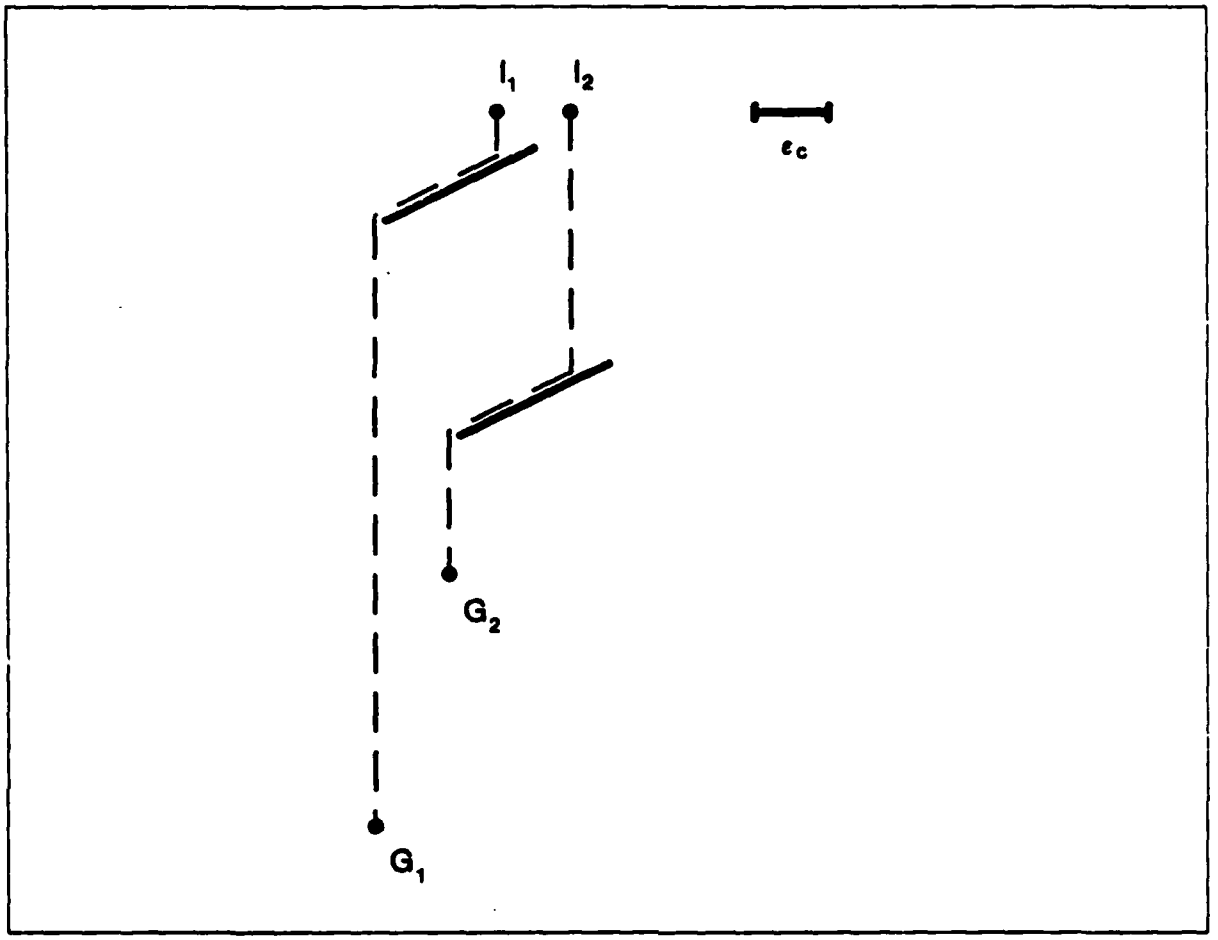


Figure 9. A problem exposing the deficiency of the stateless predicate; ϵ_c is shown, $\epsilon_v = \epsilon_\omega = 0$, $I = I_1 \bigcup I_2$, $G = G_1 \bigcup G_2$. Solution of this problem with a single motion requires a decision to terminate based on sensory information obtained earlier in the motion. The memoryless predicate requires two motions in the worst case.

In general, similar reasoning shows that we can construct subgoals on a per-motion basis. We can use a modified form of the definition of suitable subgoals: Let $\{R_{\theta,\alpha}\}$ be the set of all sets R satisfying $R = P_{\theta,R}(\{G_\alpha\})$ for some θ, where we define $P_{\theta,R}$ as:

$$P_{\theta,R}(\{G_\alpha\}) = \{c_{init} \mid \forall\, c^*_{init} \in B(c_{init})\; s_\theta \in S(c^*_{init}, R, \{G_\alpha\})\}$$

Thus, we can construct a much smaller set of conceptually simpler subgoals without affecting the performance of the approach.

Look-ahead predicate.

As observed in section II.A, to avoid any lost opportunities the termination predicate ought to terminate as soon as it can verify presence in a goal. Hence, the termination predicate constructed for FMP terminates the motion the instant that all possible trajectories are included in a common goal. It is interesting to note that this constraint can

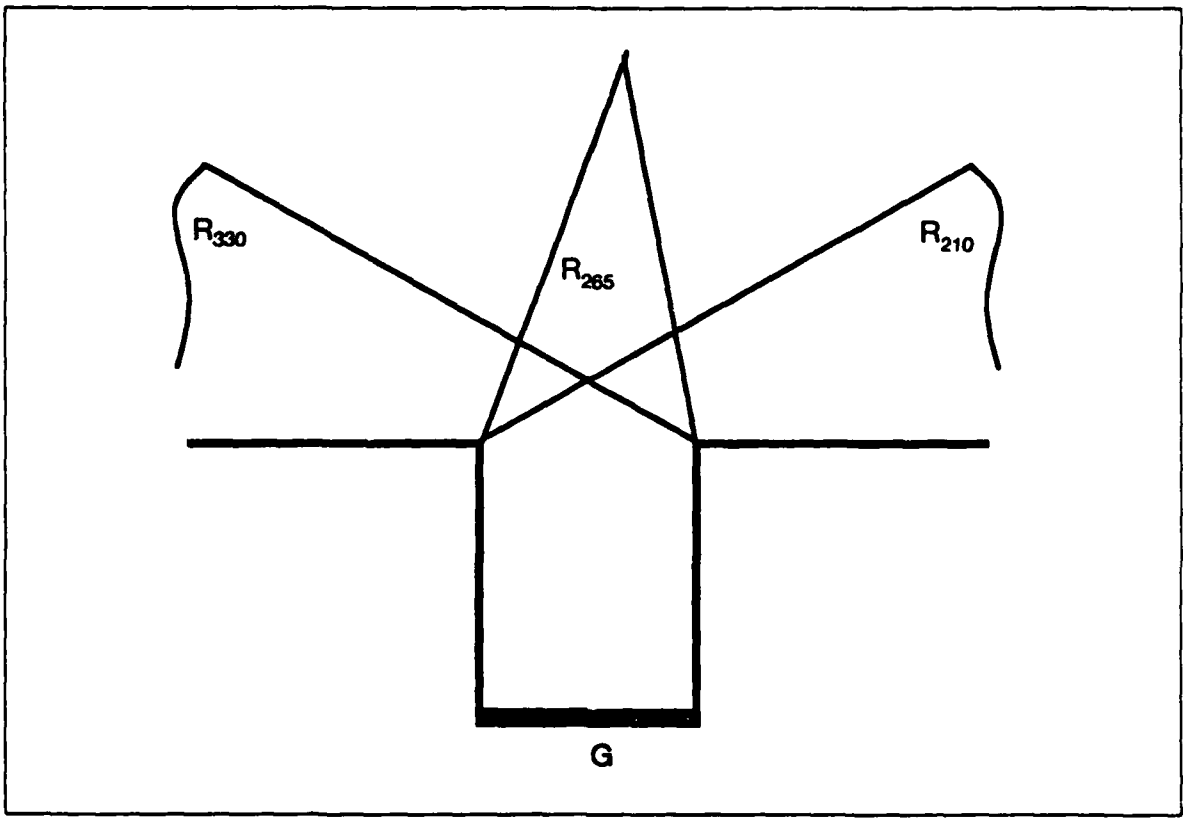


Figure 10. Simple, but sufficient, subgoals. These subgoals are not the largest subgoals satisfying $R = P_R(\{G_a\})$, but their use results in no performance compromise.

sometimes profitably be relaxed. We might allow the predicate to postpone termination if two conditions are satisfied:

1) Later opportunities to terminate are assured.

2) Termination in a "better" subgoal is possible.

Although implementation is not a primary concern of this paper, it should be noted that the mechanisms used for constructing subgoals could probably be applied here as well. We can imagine that the termination predicate would first make sure that it is not exiting the convergence region, and then consider whether any of the trajectories in the trajectory directory intersect a better subgoal.

An example is shown in figure 11. This is the usual peg in hole problem, shown as the first motion is in progress. The robot's location is constrained to lie within the uncertainty ball $B(c^*)$, which tells us that the robot is in the subgoal G^2, hence FMP's termination predicate would terminate and FMP would proceed with the second motion. However, it might be smarter to wait a little longer—by good fortune the ultimate goal G can be attained without the use of the second motion.

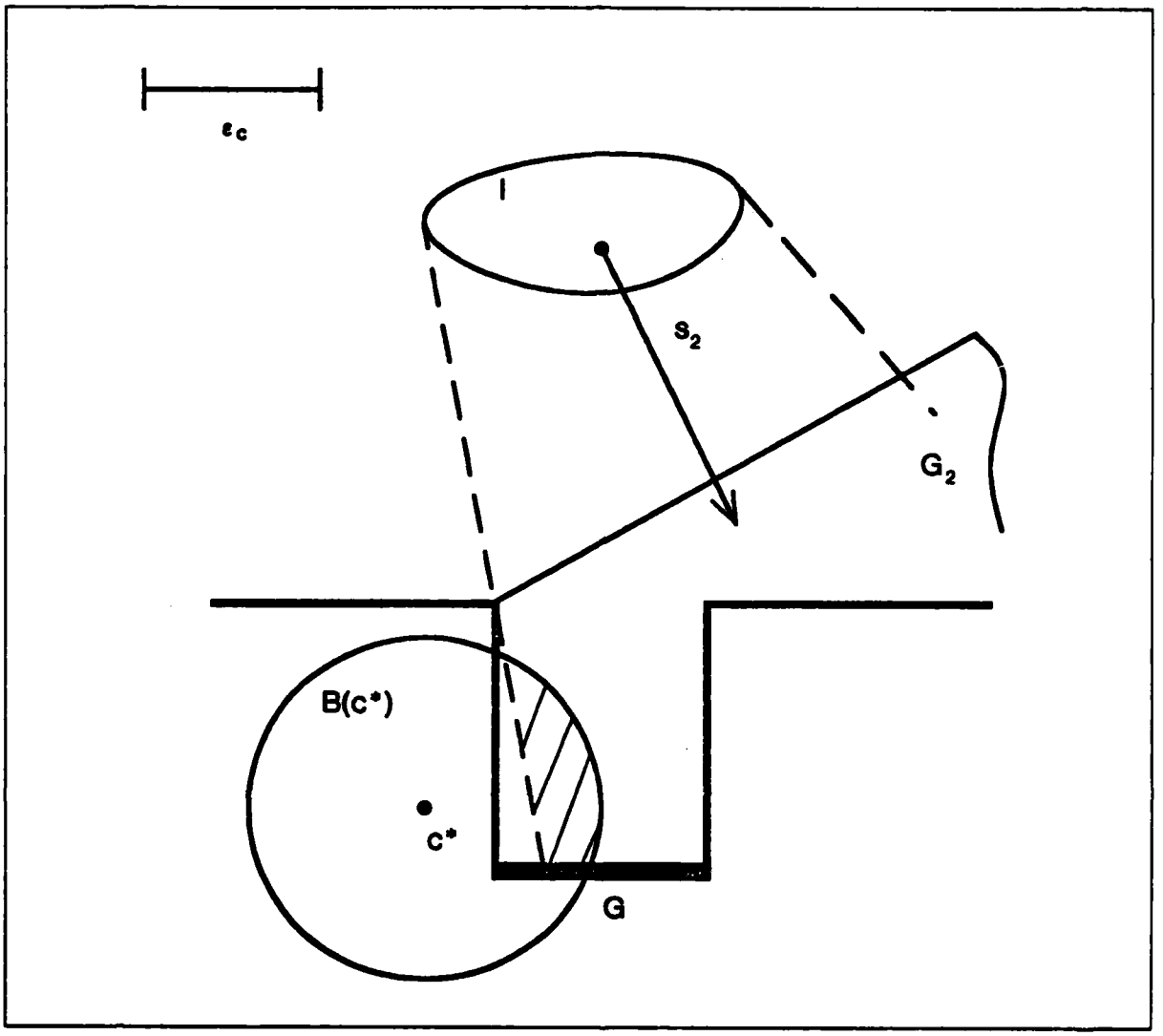


Figure 11. A simple case illustrating the value of postponing termination. The subgoal G^2 has been attained, but the ultimate goal will be attained directly if termination is postponed.

The question which naturally arises is just how much does this new predicate affect the performance of the planner. In section III we will see that FMP provides optimal worst-case performance. We have seen an example of a problem which preserves the worst-case performance but can give improved performance when the worst-case does not occur; thus FMP is not optimal in performance. Nor is the look-ahead predicate variation optimal. Improved performance could be obtained by making the choice of s from $S(c^*_{init}, R, \{G_\alpha\})$ so as to improve the chances of skipping later steps. However, our models do not have enough structure to develop such strategies. If we really wanted to pursue this problem, we would probably want to associate a probability distribution with the uncertainty regions in configuration-space.

III. Correctness and Bounded-Completeness of FMP.

A. Correctness.

Theorem: If FMP converges, i.e. if at some level a subgoal equal to the initial set I is found, then the resulting motion program will attain the goal.

Proof: We will use induction on the number of motions in the final strategy.

Basis: Let I and G be initial and goal sets for which the planning procedure converges with just one step. Then $I \in \{R_\beta\}$, i.e. I is a suitable subgoal. By definition, I satisfies the equation $I = P_I(\{G\})$. By definition of P this implies that for any $c_{init} \in I$, and any $c^*_{init} \in B(c_{init})$, $S(c^*_{init}, I, \{G\})$ is not empty. Thus the algorithm may choose a command nominal velocity s^* from $S(c^*_{init}, I, \{G\})$. By definition of S, this implies that termination will occur. We can show that the goal G is attained if we can show that the termination predicate is correct.

The correctness of the termination predicate is almost self-evident. It constructs and maintains a list of all possible robot trajectories, i.e. every trajectory consistent with control and sensory information. It only terminates if the present configuration of every possible trajectory lies within the goal G. Guaranteed termination implies a guarantee of attaining one of the immediate goals.

Induction: Assume that whenever the procedure converges in $N-1$ steps the resulting plan is correct. We must show that if it converges in N steps then the plan is correct. First we consider the first motion s^N to be executed by the robot. Using reasoning similar to the proof of the basis step, we can show that the robot will reliably attain one of the goals $G^N = R^{N-1}$. Now the remaining $N-1$ steps are equivalent to the plan produced for a new manipulation problem with $I' = R^{N-1}$ and $G' = G$. By assumption such a plan is correct, i.e. the remaining $N-1$ steps will reliably take the robot from G^N to the goal G. Thus the original N-step plan reliably attains G from I. ∎

B. Bounded-Completeness.

Theorem: Let G be the desired goal, and consider a solution to be a program which reliably attains G with a bounded number of straight, guarded, generalized-damper, motions. If a solution exists, then the synthesis procedure also attains the goal.

Proof: We will prove a stronger result: if the synthesis procedure does not converge with N motions, then no solution of length N or less exists. This means that if the procedure fails to converge in a finite number of steps, no bounded solution exists, but it also means that the synthesis procedure achieves the best worst-case performance in terms of number of motions. The proof is by induction. First we prove the result for $N = 1$.

Basis: $N = 1$. Assume that the synthesis procedure does not converge on the first motion, and consider an arbitrary motion program making only one motion. Because the synthesis procedure did not converge, we know that the set of suitable subgoals does not include all of the initial set I, i.e. $I \notin \{R_\beta\}$, which means that there is no command velocity for which our termination predicate would be guaranteed to terminate, i.e. for some c^*_{init}, $S(c^*_{init}, I, G) = \emptyset$. Of course, we are considering an arbitrary program, which is not constrained to use the termination predicate we would have constructed. However, we can show that if our termination predicate can't terminate, then no termination predicate can terminate correctly. Assuming c^*_{init} is chosen so that $S(c^*_{init}, I, G) = \emptyset$, our termination predicate terminates as soon as all trajectories consistent with control and sensory data are included in G. If our termination predicate might never terminate, it is because there is a sensor trajectory consistent with a set of trajectories which are at no time included in G. Thus there is an initial observed position c^*_{init} and a sensor trajectory such that when the motion program terminates there are feasible robot trajectories not in the goal. We conclude that the program cannot reliably attain G.

Induction step. Assume that, if the synthesis procedure does not converge in $N-1$ steps, then no solution of length $N-1$ or less exists. We will show that this holds for N as well.

Consider an arbitrary motion program of length N. Assume the synthesis procedure does not converge in N steps; then the initial set I is not in the last set of subgoals computed, i.e. $I \notin \{R^N_\beta\}$. Using the same reasoning as in the proof of the base step, we can show that the first motion s^N cannot reliably attain any of the goals $\{G^N_\alpha\}$. Let Q be the set of all possible robot positions after termination of the first motion. Since $Q \notin \{R^{N-1}_\beta\}$, the synthesis procedure does not converge in $N-1$ steps given initial set Q. By hypothesis, no motion program of length $N-1$ or less could attain G from initial set Q. We conclude that the original N-step program cannot reliably attain G from I. ∎

References

Brooks, R. A. and T. Lozano–Pérez, "A Subdivision Algorithm in Configuration Space for Findpath with Rotation," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, AI Memo 684, December, 1982 (also IJCAI–83 Proceedings).

Erdmann, M., "On a Representation of Friction in Configuration Space," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, unpublished report, January, 1983.

Inoue, H., "Force feedback in precise assembly tasks," Artificial Intelligence Laboratory, Massachusetts Institute of Technology, AIM–308, August, 1974 (Reprinted in Winston, P. H. and R. H. Brown (eds), *Artificial Intelligence: An MIT Perspective*, MIT Press, 1979).

Lozano–Pérez, T., "Automatic planning of manipulator transfer movements," *IEEE Trans. Systems, Man, Cybernetics*, vol. SMC–11, no. 10, 1981 (Reprinted in Brady, M. et. al. (eds), *Robot Motion*, MIT Press, 1983).

Lozano–Pérez, T., "Spatial planning: a configuration space approach," *IEEE Trans. Computers*, vol. C-32, no. 2, February, 1983.

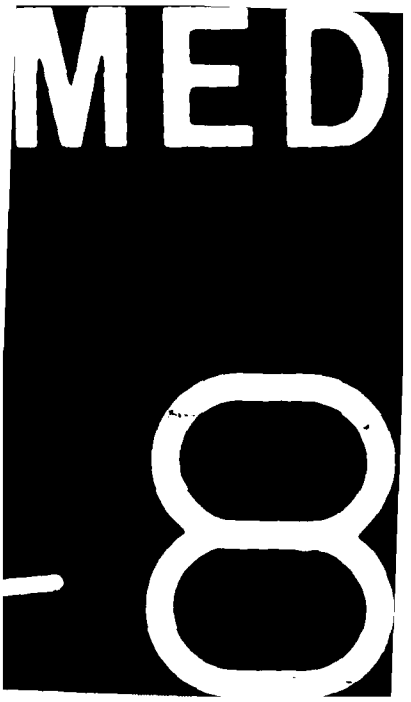
Lozano–Pérez, T., M. Mason, and R. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Robots," proceedings, International Symposium of Robotics Research, Bretton Woods, New Hampshire, August 1983.

Mason, M., "Compliance and force control for computer controlled manipulators," *IEEE Trans. Systems, Man and Cybernetics*, vol. SMC-11, no. 6, 1981 (Reprinted in Brady, M. et. al. (eds), *Robot Motion*, MIT Press, 1983.)

Mason, M., "Compliant Motion," in Brady. M. et al. (eds), *Robot Motion*, MIT Press, 1983.

Nevins, J. L., and D. E. Whitney, "The Force Vector Assembler Concept," Proceedings, First IFToMM Symposium on Theory and Practice of Robots and Manipulators, 1974.

Will, P.M., and D.D. Grossman, "An experimental system for computer controlled mechanical assembly," *IEEE Trans. Computers*, vol. C-24, no. 9, 1975.


MED
8